ARL-TR-7276 ● APR 2015

US Army Research Laboratory

Fast Computation on the Modern Battlefield

by David L Doria, Jamie K Infantolino, and Peter J Schwartz

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-TR-7276 ● APR 2015

US Army Research Laboratory

Fast Computation on the Modern Battlefield

by David L Doria, Jamie K Infantolino, and Peter J Schwartz
Computational and Information Sciences Directorate, ARL

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
April 2015	Final	NA

4. TITLE AND SUBTITLE
Fast Computation on the Modern Battlefield

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
David L Doria, Jamie K Infantolino, and Peter J Schwartz

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
US Army Research Laboratory
ATTN: RDRL-CIH-S
Aberdeen Proving Ground, MD 21005-5067

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TR-7276

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
In critical and life threatening situations faced by Soldiers on the battlefield, timely response to complex information is required. In such situations, battlefield computation can help to distill data into actionable information that can lead to better decision-making and outcomes. However, computing power is limited on the tactical edge due to size, weight, and power constraints of a Soldier's mobile device. One potential solution to this problem is to offload the computation to a more powerful computer to obtain an answer as fast as possible. However, this strategy comes at the cost of the introduction of a delay due to communication latency. It is therefore important to design the offloading mechanism intelligently. This report, presents a model to estimate the performance of offloading systems in current and future scenarios. The modularity of this model allows system designers to replace model components with the accuracy and level of detail necessary for their analyses. This report examines how this type of model is useful for making decisions about when and where to offload jobs, as well as making hardware acquisition decisions for improving the system over time.

15. SUBJECT TERMS
computation, offloading, tactical

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE	UU	20	David L Doria
Unclassified	Unclassified	Unclassified			19b. TELEPHONE NUMBER (Include area code) 410-278-2310

Contents

List of Figures

List of Tables

1. Introduction

In critical and life threatening situations faced by Soldiers on the battlefield, timely response to complex information is required. In such situations, battlefield computation can help to distill data into actionable information that can lead to better decision-making and outcomes. However, computing power has been historically limited on the tactical edge due to size, weight, and power constraints of mobile devices. One potential solution to this problem is to offload the computation to a more powerful computer to obtain an answer as fast as possible. Computation offloading provides an alternative strategy in which the user's computational job is communicated to another computing device with greater processing power that processes the data and communicates the result back to the user. The potential advantages of computation offloading include decreased computation time, decreased battery consumption due to computation, and possibly increased security and resiliency. However, these features come at the cost of the introduction of communication latency and increased battery consumption due to communication. It is therefore important to design the computation mechanism intelligently so as to maximize the advantages and minimize the disadvantages.

Another important aspect to consider is the continually evolving nature of computers and processing power. The power available today on most handheld devices is equivalent to the power available on a desktop only a few years ago. It is important to consider how far into the future computing on various devices will be possible when considering long-term planning. In this report, we present the development of a computation offloading model used to determine the best strategy to minimize total response time now and in the future.

In Section 2, we discuss related and previous work in this area. In Section 3, we detail our motivation for this work and explain why our high-level approach to modeling is an important step in the design of an offloading system. In Section 4, we describe the objectives of our model and the process we used to develop it. Section 5 details the types of analysis that are possible using a simple instantiation of the model we have described. Finally, in Section 6 we present some conclusions based on initial results and explore potential future work.

2. Related Work

As mobile technology has matured and increased in popularity, there have been many efforts to reduce energy requirements while increasing the computational power of the devices. The concept of offloading computations to nearby or even

distance resources has been widely explored. Previous studies[1,2] have shown certain portions of the code are offloaded to increase overall performance both in terms of increased computation power and decreased energy consumption. This shows that offloading is feasible in various situations; however, this work does not concentrate on the time sensitive analysis that is needed on the modern battlefield.

With computation offloading comes an overhead cost of transferring data to the remote device and transferring the answer back. Therefore, the benefits of computing on other devices must outweigh the data transfer time. The total cost (transfer time plus computation time) has been explored.[3] There have been studies[4] that concentrated on minimizing energy consumption as the main objective for offloading. This is important to consider in a time critical scenario, but it cannot be the sole consideration. Obtaining the computational results as quickly as possible while maintaining lower energy consumption is often a useful strategy.

Another idea that has been explored is to provide a software-based framework that will offload portions of the application to any available hardware.[5,6,7] Each of these provides a framework or software interface that will offload various modules within the application to the cloud without the programmers explicitly specification which is different than previously discussed work because the software based framework makes the offloading decisions. However, this strategy does not predict the potential performance gained and there is no guarantee of the performance that will be achieved.

Delegation is another computation strategy used by mobile application developers to decrease computation time. In another study,[8] the authors compare offloading to delegation to determine the benefits and shortcomings of each in different scenarios with current technology. It was determined that offloading is more feasible with current mobile technology, which is vital for success on the battlefield.

The use of a cloudlet (or collection of mobile devices with cloud-like services) for computation offloading has been studied[9,10] and shown to reduce energy costs while maintaining an acceptable computation rate. Under this strategy, high-powered computers can be geographically farther apart and reserved exclusively for very computationally intensive applications. This reduces energy costs overall; however, the acquisition and deployment of additional physical hardware may be necessary.

3. Motivation

The purpose of this work is to develop a set of models that we will use to evaluate and compare alternative computation strategies at a conceptual level. These models have several purposes. First, they motivate and quantify the need for computation offloading, particularly in tactical environments. They also allow rapid evaluation of alternative computation strategies under a variety of conditions. Our models can also be applied to time-critical applications in nonmilitary settings, such as emergency response.

For all of the computation strategies considered in this work, we are interested in analyzing the total response time, *r* of the system, which we define as the time it takes to provide a response to the user. Total response time can be computed as the sum of the communication time and the computation time. We define communication time as the time that it takes to transfer the required data to the device that performs the computation and to transfer the resultant data back to the user. We define computation time as simply the time required for the target device to process the data.

In any setting, performing computation solely on the user's handheld device will require no communication time, but relatively high-computation time. In a commercial setting (with high-bandwidth network infrastructure), data can be sent very quickly to a remote data center with very high-computational capacity, so computation time can be reduced dramatically at the cost of only a relatively small increase in communication time. While this communication latency might be intolerable for real-time interactive applications (e.g., augmented reality), it is acceptable for many common applications (e.g., route planning). As long as total response time is acceptable, monetary cost minimization is the dominant factor in designing a commercial computation offloading system.

In a military setting, the network infrastructure is often disrupted, intermittent, and low-bandwidth. This greatly increases communication time, which opens a potential technology gap. It might be possible to fill this gap by deploying strategic (regional), tactical (local)[11], or mobile (1-hop) High-Performance Computers (HPCs) within the network providing new offloading targets between the centralized data center and the user's handheld device. We illustrate this gap visually in Fig. 1.

As shown in Fig. 1, as the distance between the user's handheld device and the device that performs the computation increases (in terms of number of network hops), there are 2 important changes. First, the communication time increases because the data must be sent farther through the network. Second, the

computation time decreases because the data can reach a more powerful computer. Based on these ideas, in this work we develop a model that combines existing models of computer and network performance along with reasonable system architecture assumptions and parameter estimates.

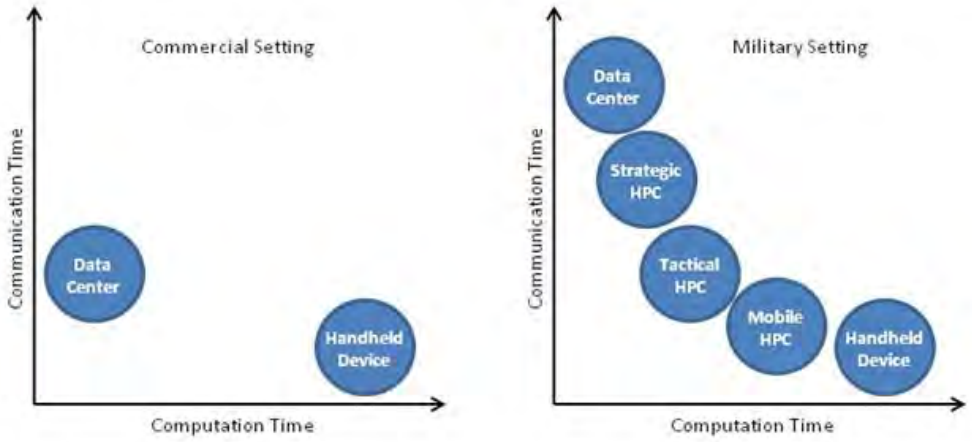

Fig. 1 Conceptual sketch of various regions in the space of computation strategies

4. Computation Offloading Model

When designing our model, our 2 major goals were simplicity and modularity. We wanted the model to not necessarily answer every question for every scenario, but rather expose easy to adjust values that allow designers to model their situation of interest. By designing the model modularly, we minimize the interactions between the various components, localizing the effects of a change to any one component.

Our model describes the relationships between many components, including the computational job, the network, the available compute devices, the computation strategy, utility to the user, and the development of technology over time. The Computational Job characterizes the computation that the user wishes to perform. A set of Compute Devices describes computers that are potentially available to execute the Computational Job. The Network describes the communication network that can transfer data between the user's handheld device and other Compute Devices.

4.1 Computational Job

We describe a computational job with several parameters: the number of operations necessary to execute the job, the deadline by which the user needs a response, and the max

utility, u_{max}, that is achieved for providing a response to the user before the deadline, d, the size of the required data to transfer, and the fraction of operations that can be executed in parallel.

4.2 Utility Function

The system response time is sent to the Utility Function, which calculates the utility to the end user based on the parameters of the Computational Job. While intuitively a shorter total response time seems like it would always be preferred, in some cases a faster response may not be beneficial. For example, in a mission planning scenario, the solution may be required by a specific time. As long as the solution is provided before that time, it has the same utility to the user. After that time, however, the solution is not useful, as a decision would have had to be made using an alternative method. As such, defining a utility function, $u(t)$, is a critical step in analyzing the affect of tradeoffs of system parameters (Fig. 2).

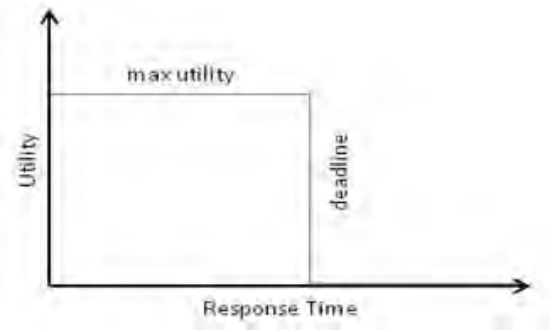

Fig. 2 Utility step function

In the mission planning situation described above, the utility function is a step function. That is, if the response time is within the deadline, then the system has provided the max utility. Otherwise, the system has provided 0 utility. This simple utility function can be written as

$$u = \begin{cases} u_{max}, & r \leq d \\ 0, & otherwise \end{cases} . \quad (1)$$

Of course, a system designer should customize the Utility Function by specifying a more complex function of response time (multiple steps, piecewise linear, exponential, etc.) to accurately reflect the scenario at hand.

4.3 System Latency

As described previously, the 2 major components of the model are communication time and computation time. For simplicity of exposition, in this work we adopt the most naive definition of communication latency, l_{comm}, which we define to be

$$l_{comm}\left(f, h, BW_y, y\right) = \sum_{i=1}^{h} \frac{f}{BW_y[i]} = \sum_{i=1}^{h} \frac{f}{BW_0[i]1.5^y}, \quad (2)$$

where f is the file size of the data necessary to transfer, BW_0 is the average bandwidth of the h^{th} hop of the channel between the user and the target computation device in the current year, and y is the number of years in the future. This value is a good indicator of the performance of a current system, but also can help determine the best hardware acquisitions for future systems. Again, because of the modularity of the model, it is possible for designers to substitute the most accurate value for communication latency that can be obtained in their system.

It is necessary to use a slightly less naive model of computation to observe realistic effects. We compute the computation latency as

$$l_{comp} = \frac{O}{F \cdot S}, \quad (3)$$

where O is the total number of operations, F is the number of FLOPS per processor, and, from Amdahl's law[12], the theoretical speedup of multithreaded execution, S, can be computed as

$$S = \frac{1}{(1-p)+\frac{p}{n_p}}. \quad (4)$$

Here, p is the proportion of the application that is parallelizable. To model the total system latency, we add the communication and computation latencies to obtain

$$l\left(f, O, p, h, BW_0, n_p, F_0, y\right) = l_{comm}\left(f, h, BW_y, y\right) + l_{comp}\left(O, p, n_p, F_0, y\right). \quad (5)$$

Equation 5 is the main contribution of this work. It clearly lays out how an offloading system designer can substitute known values of their hardware and applications to obtain a general idea of system performance.

4.4 Usefulness Over Time

Figure 3 illustrates how the development of technology over time is captured by our model. The thick blue lines represent the development of technologies related to the Network (e.g., increase in bandwidth), and Compute Devices (e.g., increase in FLOPS). By repeatedly applying these transformations, we can attempt to predict how user utility will change in the future. The computation strategy that performs best with today's technology might not be the same computation mechanism that performs best with the technology that will be fielded 5 to 10 years from now. This helps designers identify which strategy is most worthwhile to pursue for lasting effectiveness.

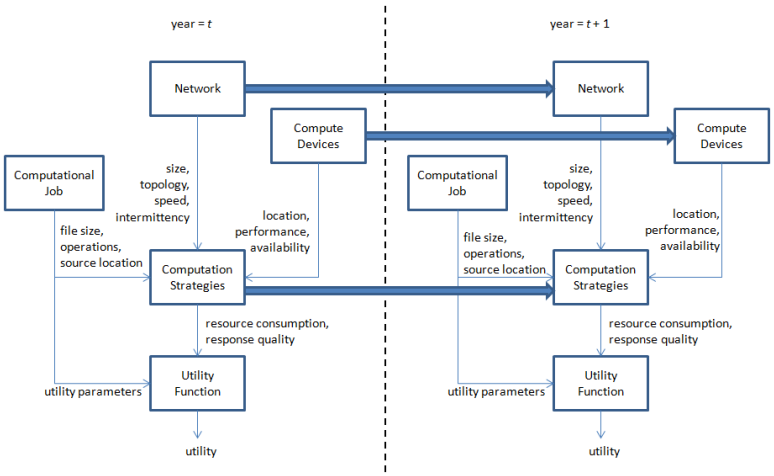

Fig. 3 An illustration of the relationships between the major model components. The moving from left to right over the dashed vertical line indicates a step 1 year forward in time

5. Analysis

To demonstrate how a system designer can use our model, we illustrate the simplest computation strategy to not offload the computational job, but to simply execute it on the user's handheld device. Since no data is being sent over the network, there is no need to model any network communication. The user's handheld device contains a single processor, so there is no opportunity for parallel execution. We take Utility to be a step function. The deadline and max utility are provided as model inputs that are part of the Computational Job.

In the following table, we provide some realistic sample values that we will use to demonstrate the application of our model. In Fig. 4, we plot the communication and computation latency, as well as the derived total response time as a function of distance from the user using the values in the table.

Table Estimated values versus distance

Distance From User (m)[a]	Hops From User	Number of Processors	Bandwidth (Mbps)[b]
10	1	8	0.5
100	2	64	0.5
1000	3	512	40
10000	4	4096	40
100000	5	32768	200

[a]m, meters; [b]Mbps, megabits per second.

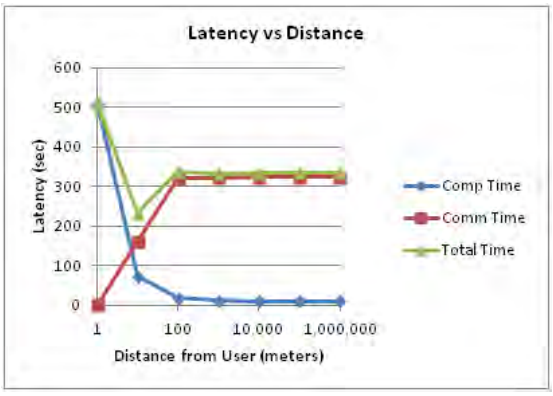

Fig. 4 Prediction of communication, computation, and total response time vs. distance from user

In this example, the user will be executing a job where the number of operations needed to complete the job is provided as a model input that is part of the Computational Job. According to Moore's law[13], transistor counts double every 18 months, which corresponds to an increase of 60% per year. We take this value for our FLOPS change over time. We set FLOPS at time 0 (the year 2014) to 2.5 billion (based on the 2.5-GHz clock speed of a standard smart phone, and assumes 1 operation per clock cycle), and arrive at the expression for the improvement of the user's computation capability over time as $FLOP\ S(t) = FLOP\ S_{t_0} 1.6^t$. With these assumptions, we see in Fig. 5 that by simply waiting 3 years, the user can obtain considerable more utility without changing any other system parameters.

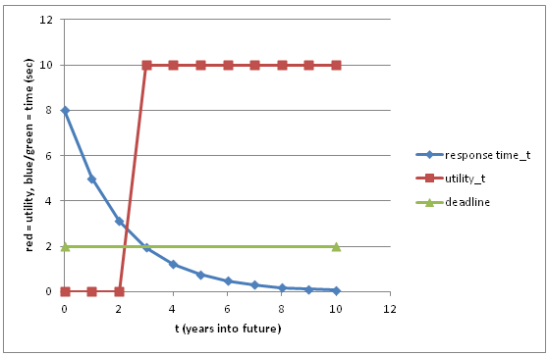

Fig. 5 Response time and utility as a function of years into the future

Decision Boundaries

One of the most powerful ways that our model can be employed is to make a decision as to where to offload a computational job. After a system designer specifies all of the model parameters corresponding to their scenario, a chart, as shown in Fig. 6, can be produced to give an "at-a-glance" view of the behavior of the system where the decision of where to offload any job can be read immediately. To generate these decisions, we choose the offloading target with the minimum system response time. That is, we can find the ideal target computation device by optimizing:

$$h^* = \min r(h) \tag{6}$$

where h^* is the number of hops from the user where the ideal offload target resides, and $r(h)$ is the system response time for a device h hops away from the user. In Fig. 6, we show the decision boundaries for the system described by our running example.

Parallelizable Operations (billions) \ File Size (MB)	1	4	16	64	256	1024	4096	16384	65536	262144	1048576
1048576	49.13	51.37	60.30	96.02	238.93	806.77	2892.03	8990.52	26011.67	84265.90	207813.39
262144	42.84	45.07	54.01	89.73	231.69	753.01	2277.63	6532.92	21096.47	51983.35	104897.60
65536	41.27	43.50	52.43	87.92	218.25	599.41	1663.23	5304.12	13025.84	26254.40	26254.40
16384	40.88	43.11	51.98	84.56	179.85	445.81	1356.03	3286.46	6593.60	6593.60	6593.60
4096	40.78	43.00	51.14	74.96	141.45	369.01	851.61	1678.40	1678.40	1678.40	1678.40
1024	40.75	42.79	48.74	65.36	122.25	242.90	449.60	449.60	449.60	449.60	449.60
256	40.70	42.19	46.34	60.56	90.73	142.40	142.40	142.40	142.40	142.40	142.40
64	40.55	41.59	45.14	52.68	65.60	65.60	65.60	65.60	65.60	65.60	65.60
16	40.40	41.29	43.17	46.40	46.40	46.40	46.40	46.40	46.40	46.40	46.40
4	40.32	40.79	41.60	41.60	41.60	41.60	41.60	41.60	41.60	41.60	41.60
1	40.20	40.40	40.40	40.40	40.40	40.40	40.40	40.40	40.40	40.40	40.40
Color code	0 hops		1 hop		2 hops		3 hops		4 hops	5 hops	6 hops

Fig. 6 Offloading decision boundaries

In Fig. 6 each cell corresponds to a computational job with different attributes. The color of the cell indicates the optimal offloading decision that should be made for a job with the corresponding attributes. The number of serial operations is held constant over all jobs. We noticed clear "bands" of job attributes where offloading to a particular place in the network is optimal. A system designer can use this type of output to quickly see what the effect of changing some system parameters, like the number of processors placed at a particular point in the network, would have on the overall system response time for particular job types.

6. Conclusions and Future Work

We have presented a simple, modular model of computation on the modern battlefield. We have shown that a high-level model is useful for studying a number of important properties of a system designed to provide computational assistance to an end user. By customizing such a model with the existing or proposed parameters of an offloading system, system designers can make quick, intelligent decisions about where to place resources, and how to best take advantage of them.

In future work, we will compute error measurements as compared to fielded systems. We will then perform a sensitivity analysis to determine the accuracy of our model, as well as determine which parameters the model is most sensitive to, informing us as to which parts of the model should be improved first.

7. References

1. Kocyigit A, Eren PE, Kaya M. A mobile computing framework based on adaptive mobile code offloading. In Software Engineering and Advanced Applications; Aug 2014.
2. Puccinelli D, Giordano S, Ferrari A. Code offloading on opportunistic computing. In IEEE International Conference on Pervasive Computing and Communications Workshops; Mar 2014.
3. Samaan N, Barrameda J. A novel application model and an offloading mechanism for efficient mobile computing. In IEEE International Conference Wireless and Mobile Computing, Networking and Communications; Oct 2014.
4. Mtibaa A, Afnan F, Harras KA. Towards computational offloading in mobile device clouds. In 2013 IEEE International Conference on Cloud Computing Technology and Science.
5. Balasubramanian A, Cho DK, Wolman A, Saroiu S, Chandra R, Bahl P, Cuervo E. Maui: making smartphones last longer with code offload. In Proceedings of the International Conference on Mobile Systems, Applications, and Services; June 2010.
6. Ihm S, Maniatis P, Naik M, Chun BG. Clonecloud: boosting mobile device applications through cloud clone execution. In Proceedings of the Sixth Conference on Computer Systems; Salzburg, Austria, Apr 10–13, 2011.
7. Pascual-Iserte A, Vidal JMO. Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading. IEEE Transactions on Vehicular Technology. 2014;(99).
8. Satish NS, Rajkumar B. Computational offloading or data binding? Bridging the cloud infrastructure to the proximity of the mobile user. In IEEE International Conference on Mobile Cloud Computing, Services, and Engineering. 2014.
9. Gani-A SM. Mobile cloud computing: critical analysis of application deployment in virtual machines. In Proceedings of the International Conference on Information and Computer Networks. Feb 2012.
10. He L, Liu L, Li K, Jarvis SA, Gao B. From mobiles to clouds: developing energy-aware offloading strategies for workflows. In Proceedings of the ACM/IEEE International Conference on Grid Computing. Sep 2012.

11. Shires D, Henz B, Park S, Clarke J. Cloudlet seeding: spatial deployment for high performance tactical clouds. In Parallel and Distributed Processing Techniques and Applications. 2012.

12. Amdahl GM. Validity of the single-processor approach to achieving large scale computing capabilities. In Proceedings of the American Federation of Information Processing Societies Conference. 1967.

13. Moore GE. Progress in digital electronics. In Technical Digest of the International Electron Devices Meeting. 1975.

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
2 (PDF)	DIRECTOR US ARMY RESEARCH LAB RDRL CIO LL IMAL HRA MAIL & RECORDS MGMT
1 (PDF)	GOVT PRINTG OFC A MALHOTRA
1 (PDF)	DIR USARL RDRL CIH S D DORIA

INTENTIONALLY LEFT BLANK.